Lesson 13
Program implementation in Pub/Sub mode for the MQTT clients and MQTT broker using Eclipse Paho

Program implementation in Pub/Sub mode

- Eclipse Paho for Program implementation for the MQTT clients and MQTT broker
- MQTTclient, unlike HTTP client, does not have to pull the messages (commands) which the device needs for its control
- A MQTT broker pushes the messages to the client, provided a client
- has subscribed to those

MQTT Broker

- A broker functions like a dashboard, where the messages first reach from the sources and from where the messages then dispatch to the subscribers (ones who makes a request for dispatching messages)
- Each MQTT client may be required to use an *always connected UDP (or TCP) connection* with a broker.

The MQTT broker

- Uses buffers for all messages
- This facilitates the dispatch them when a client reconnects after an interruption
- An interruption is always a possibility in ROLL environment

Example of MQTT Pub/Sub connectivity between devices and application layers

- Connected Streetlights publishing the sensed data and subscribing for the control data
- Using the MQTT clients during communication with the application layer

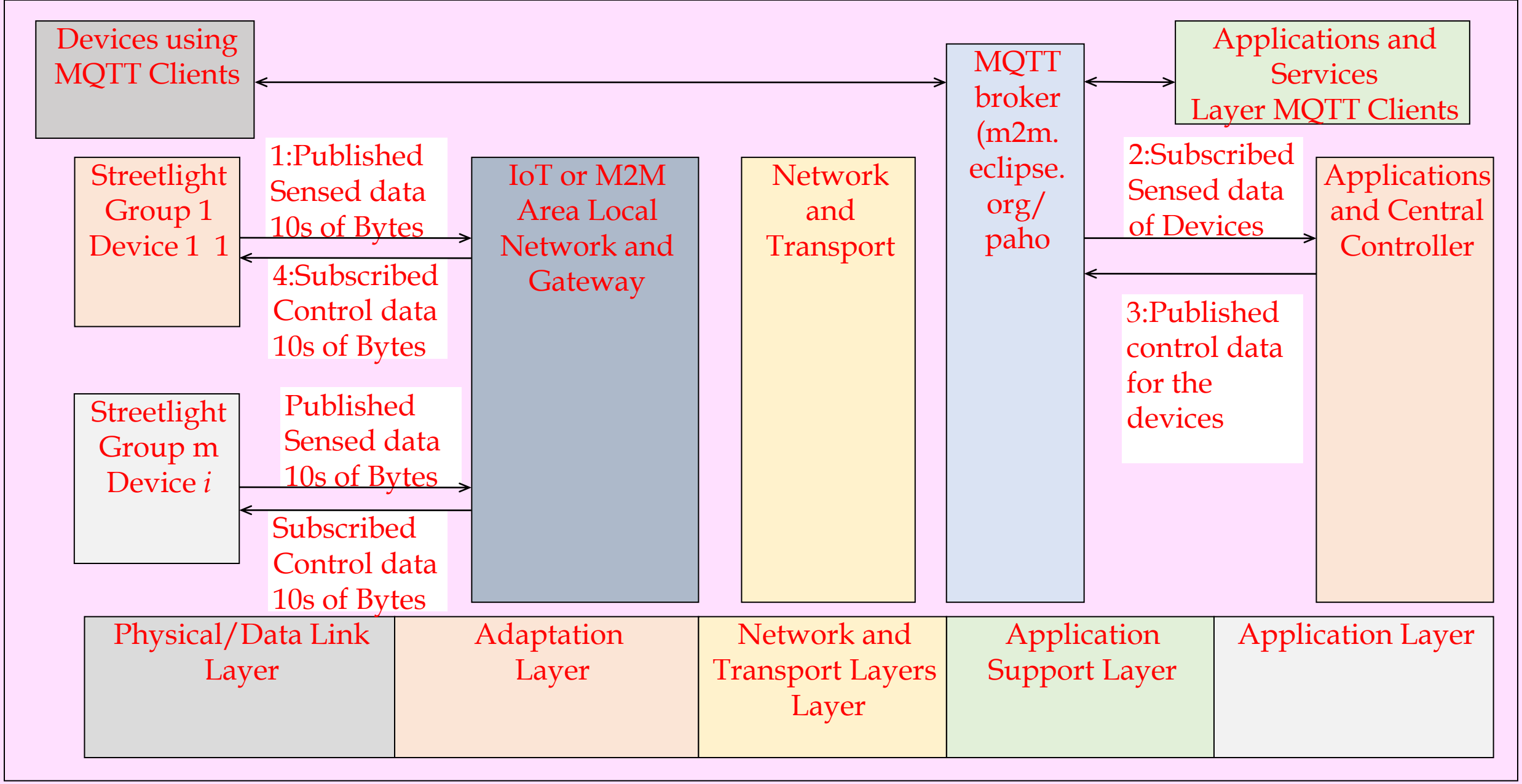

Fig. 9.2 Connected Streetlights publishing the sensed data and subscribing for the control data using MQTT clients' communication with Application layer MQTT clients through an MQTT Broker

Example 9.11 Eclipse Paho Java implementation of MQTT clients-broker architecture

- Communication of sensed data to the applications and central controller for the control of streetlights

Step (i)

- Import Java packages for MQTT clients, broker and applications and the Publisher and Subscriber classes for constructing the objects for publishing sensed data of the devices, such as streetlights
- Package org.eclipse.pahoIoTStreetLights;
- import org.eclipses.paho.client.mqttv3.MqttClient;
- import org.eclipses.paho.client.mqttv3.MqttException;
- import org.eclipses.paho.client.mqttv3.MqttMessage

Step (ii)

- (ii) Implement the codes for Publisher class for control data from the central controller and sensed data
- from the devices.

Step (iii)

- (iii) Implement the codes for Subscriber class for subscribing of sensed data by the application/central Controller.

Step (iv)

- (iv) Implement the codes for Subscriber class for subscribing of control data by the device clients

Step (v)

- (v) Implement the codes for observing ‘Last Will and Testament’ (LWT) message on disconnection/failure of communication of messages on topic sensed data and control data and communicating again on reconnection using MQTT broker

Step (vi)

- (vi) Implement the codes for threads for publishing the messages for the sensed data, for example, for traffic presence and lighting needs in Internet of Streetlights

Coding Steps

- Write codes
- (i) Publisher class for constructing the objects for publishing sensed data of the devices, such as streetlights

  Construct mqttClient with mqqt broker, client ID inputs
- (ii) Let next step be use MAC Address as Application ID for subscriber ID and the initialisation of the MqttClient instance almost the same as above publisher above code, except that use-pubApp as a suffix for the Client_Id.

Coding Steps

- (iii) Subscriber class for subscribing of sensed data by the application/central controller
- Let next step be use MAC Address as Application ID for subscriber ID and the initialisation of a MqttClient instance is as per publisher codes, except that use -subApp as a suffix for the Application_Id

Coding Steps for (iii)

- public class Subscriber {
- public static final String MQTT_BROKER_URL = “tcp://broker.mqttdashboard.
- com:1883”;
- /* client is the instance of MqttClient */
- Details in the Text of Example 9.11

Coding Steps

- (iv) Subscriber class for subscribing of control data by the device clients
- Let the MAC Address be used as Client_ID for subscriber ID and the initialisation of the MqttClient instance is almost the same as above subscriber code, except use -subDevice as a suffix for the Client_Id

Coding Steps

- (v) A disconnection is feasible. Broker needs to detect disconnection by a client. If the clientDevice has set a subscription to a message for a topic (for example SensedData), then when Device reconnect
- Broker sends ‘LWT’ message string to the topic ‘SensedData’ on reconnection. Similarly, If the clientApplication has set a subscription to a message for a topic (for example, ControlData), then when Application reconnects, Broker sends ‘LWT’ message string to the topic ‘ControlData’ on reconnection

Coding Steps

- (vi) Declaring threads for publishing the messages on the topic sensed data
- A client publishes the sensed data
- Declaring the threads for the topic and messages of each TOPIC which the client publishes from each streetlight device platform

Coding Step (vii)

- (vii) (a) Publishing the messages by clients at each streetlight on the topic sensed data
- /*Next step is publishing of messages by the clients at each streetlight device platform. Message string publishes for lightneed, working status and traffic-presence from each streetlight in each group of streetlights.*/
- Example: private void publishLightNeed() throws MqttException

Coding Step (vii)

- (b) Publishing the messages by application/central controller on the topic ‘control data’
- /*Next step is publishing of messages by the Application for each
- streetlight device platform. Message string publishes for control message
- for each streetlight in each group of streetlights.*/

Coding Step (vii)

- (c) Client application subscription of the messages on the topic sensed data
- A coding step is codes for a client subscription, which is reading the values on the topic SensedData.
- Each client device sends on the topic SensedData publishes three messages to the Broker.
- “SensedData /lightneed” ,
- “SensedData /workingstatus” and
- “SensedData/trafficpresence”

Coding Step (vii)

- (d) Use a wildcard ‘#’ instead of subscription to 3 different topics, namely, TOPIC_LIGHTNEED, TOPIC_WORKINGSTATUS, TOPIC_TRAFFICPRESENCE

Summary

We learnt

- An example for the usages of Paho for streetlights IoT
- Streetlight devices publish the sensed data using Paho Java MQTT clients
- Applications publish control data and subscribe to the sensed data
- An Eclipse Paho MQTT broker provides a gateway between the devices and applications

End of Lesson 13 on Program implementation in Pub/Sub mode for the MQTT clients and MQTT broker using Eclipse Paho